# Brain Stroke Medical Data Analysis Using Matplotlib and Seaborn

Risad Raihan Malik

# Brain Stroke Medical Data Analysis Using Matplotlib and Seaborn

I'm thrilled to share the successful completion of a groundbreaking Brain Stroke Analysis project! Here are the key highlights of my work:

🔍 Null Value Handling: Identified and meticulously addressed null values within the dataset to ensure impeccable data integrity and accuracy, laying a robust foundation for further analysis.

📈 In-depth Analysis: Conducted a thorough and insightful analysis to decipher patterns and trends related to brain stroke occurrences, providing valuable insights into this critical healthcare domain.

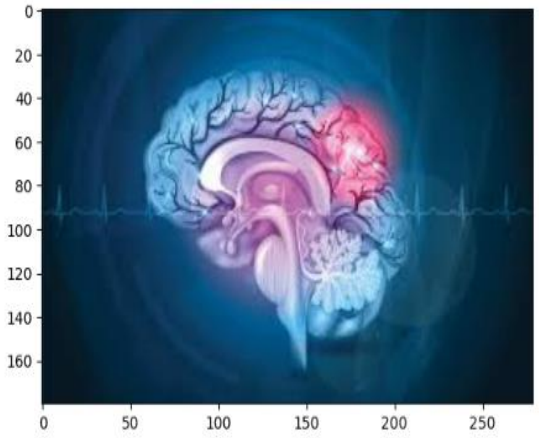
🖊 Data Manipulation Expertise: Employed advanced data manipulation techniques to fill missing values and optimize data quality, enhancing the reliability and usefulness of the dataset.

🚀 Python Libraries Mastery: Leveraged powerful Python libraries including Pandas and NumPy for seamless data preprocessing and cleaning, streamlining the analysis process and maximizing efficiency.

📊 Visualization Proficiency: Utilized state-of-the-art visualization tools such as Matplotlib and Seaborn to craft visually engaging charts and graphs, effectively communicating complex insights to stakeholders.

In [1]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

img = mpimg.imread('images.JFIF')
plt.imshow(img)
plt.show()
```



## Import Library

In [2]:
```
import pandas as pd
```

In [3]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

## Uploading Csv fle

In [4]:
```
df = pd.read_csv("healthcare-dataset-stroke-data.csv")
df
```

Out[4]:

| | id | gender | age | hypertension | heart_disease | ever_married | work_type | Residence_type | avg_glucose_level | bmi | smoking_status | stroke |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 9046 | Male | 67.0 | 0 | 1 | Yes | Private | Urban | 228.69 | 36.6 | formerly smoked | 1 |
| 1 | 51676 | Female | 61.0 | 0 | 0 | Yes | Self-employed | Rural | 202.21 | NaN | never smoked | 1 |
| 2 | 31112 | Male | 80.0 | 0 | 1 | Yes | Private | Rural | 105.92 | 32.5 | never smoked | 1 |
| 3 | 60182 | Female | 49.0 | 0 | 0 | Yes | Private | Urban | 171.23 | 34.4 | smokes | 1 |
| 4 | 1665 | Female | 79.0 | 1 | 0 | Yes | Self-employed | Rural | 174.12 | 24.0 | never smoked | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 5105 | 18234 | Female | 80.0 | 1 | 0 | Yes | Private | Urban | 83.75 | NaN | never smoked | 0 |
| 5106 | 44873 | Female | 81.0 | 0 | 0 | Yes | Self-employed | Urban | 125.20 | 40.0 | never smoked | 0 |
| 5107 | 19723 | Female | 35.0 | 0 | 0 | Yes | Self-employed | Rural | 82.99 | 30.6 | never smoked | 0 |
| 5108 | 37544 | Male | 51.0 | 0 | 0 | Yes | Private | Rural | 166.29 | 25.6 | formerly smoked | 0 |
| 5109 | 44679 | Female | 44.0 | 0 | 0 | Yes | Govt_job | Urban | 85.28 | 26.2 | Unknown | 0 |

5110 rows × 12 columns

## Data Preprocessing

head is used show to the By default = 5 rows in the dataset

In [5]:
```
df.head(10)
```

Out[5]:

| | id | gender | age | hypertension | heart_disease | ever_married | work_type | Residence_type | avg_glucose_level | bmi | smoking_status | stroke |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 9046 | Male | 67.0 | 0 | 1 | Yes | Private | Urban | 228.69 | 36.6 | formerly smoked | 1 |
| 1 | 51676 | Female | 61.0 | 0 | 0 | Yes | Self-employed | Rural | 202.21 | NaN | never smoked | 1 |
| 2 | 31112 | Male | 80.0 | 0 | 1 | Yes | Private | Rural | 105.92 | 32.5 | never smoked | 1 |
| 3 | 60182 | Female | 49.0 | 0 | 0 | Yes | Private | Urban | 171.23 | 34.4 | smokes | 1 |
| 4 | 1665 | Female | 79.0 | 1 | 0 | Yes | Self-employed | Rural | 174.12 | 24.0 | never smoked | 1 |
| 5 | 56669 | Male | 81.0 | 0 | 0 | Yes | Private | Urban | 186.21 | 29.0 | formerly smoked | 1 |
| 6 | 53882 | Male | 74.0 | 1 | 1 | Yes | Private | Rural | 70.09 | 27.4 | never smoked | 1 |
| 7 | 10434 | Female | 69.0 | 0 | 0 | No | Private | Urban | 94.39 | 22.8 | never smoked | 1 |
| 8 | 27419 | Female | 59.0 | 0 | 0 | Yes | Private | Rural | 76.15 | NaN | Unknown | 1 |
| 9 | 60491 | Female | 78.0 | 0 | 0 | Yes | Private | Urban | 58.57 | 24.2 | Unknown | 1 |

In [6]:
```
df.tail(10)
```

Out[6]:

| | id | gender | age | hypertension | heart_disease | ever_married | work_type | Residence_type | avg_glucose_level | bmi | smoking_status | stroke |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **5100** | 68398 | Male | 82.0 | 1 | 0 | Yes | Self-employed | Rural | 71.97 | 28.3 | never smoked | 0 |
| **5101** | 36901 | Female | 45.0 | 0 | 0 | Yes | Private | Urban | 97.95 | 24.5 | Unknown | 0 |
| **5102** | 45010 | Female | 57.0 | 0 | 0 | Yes | Private | Rural | 77.93 | 21.7 | never smoked | 0 |
| **5103** | 22127 | Female | 18.0 | 0 | 0 | No | Private | Urban | 82.85 | 46.9 | Unknown | 0 |
| **5104** | 14180 | Female | 13.0 | 0 | 0 | No | children | Rural | 103.08 | 18.6 | Unknown | 0 |
| **5105** | 18234 | Female | 80.0 | 1 | 0 | Yes | Private | Urban | 83.75 | NaN | never smoked | 0 |
| **5106** | 44873 | Female | 81.0 | 0 | 0 | Yes | Self-employed | Urban | 125.20 | 40.0 | never smoked | 0 |
| **5107** | 19723 | Female | 35.0 | 0 | 0 | Yes | Self-employed | Rural | 82.99 | 30.6 | never smoked | 0 |
| **5108** | 37544 | Male | 51.0 | 0 | 0 | Yes | Private | Rural | 166.29 | 25.6 | formerly smoked | 0 |
| **5109** | 44679 | Female | 44.0 | 0 | 0 | Yes | Govt_job | Urban | 85.28 | 26.2 | Unknown | 0 |

It show the total no of rows & Column in the dataset

In [7]:
```
df.shape
```

Out[7]:
```
(5110, 12)
```

It show the no of each Column

In [8]:
```
df.columns
```

Out[8]:
```
Index(['id', 'gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

This Attribute show the data type of each column

In [9]:
```
df.dtypes
```

Out[9]:
```
id                     int64
gender                object
age                  float64
hypertension           int64
heart_disease          int64
ever_married          object
work_type             object
Residence_type        object
avg_glucose_level    float64
bmi                  float64
smoking_status        object
stroke                 int64
dtype: object
```

In a column, It show the unique value of specific column.

In [10]:
```
df["work_type"].unique()
```

Out[10]:
```
array(['Private', 'Self-employed', 'Govt_job', 'children', 'Never_worked'],
      dtype=object)
```

It will show the total no of unque value from whole data frame

In [11]:
```
df.nunique()
```

Out[11]:
```
id                   5110
gender                  3
age                   104
hypertension            2
heart_disease           2
ever_married            2
work_type               5
Residence_type          2
avg_glucose_level    3979
bmi                   418
smoking_status          4
stroke                  2
dtype: int64
```

It show the Count, mean , median etc

In [12]:
```
df.describe()
```

Out[12]:

| | id | age | hypertension | heart_disease | avg_glucose_level | bmi | stroke |
|---|---|---|---|---|---|---|---|
| **count** | 5110.000000 | 5110.000000 | 5110.000000 | 5110.000000 | 5110.000000 | 4909.000000 | 5110.000000 |
| **mean** | 36517.829354 | 43.226614 | 0.097456 | 0.054012 | 106.147677 | 28.893237 | 0.048728 |
| **std** | 21161.721625 | 22.612647 | 0.296607 | 0.226063 | 45.283560 | 7.854067 | 0.215320 |
| **min** | 67.000000 | 0.080000 | 0.000000 | 0.000000 | 55.120000 | 10.300000 | 0.000000 |
| **25%** | 17741.250000 | 25.000000 | 0.000000 | 0.000000 | 77.245000 | 23.500000 | 0.000000 |
| **50%** | 36932.000000 | 45.000000 | 0.000000 | 0.000000 | 91.885000 | 28.100000 | 0.000000 |
| **75%** | 54682.000000 | 61.000000 | 0.000000 | 0.000000 | 114.090000 | 33.100000 | 0.000000 |
| **max** | 72940.000000 | 82.000000 | 1.000000 | 1.000000 | 271.740000 | 97.600000 | 1.000000 |

It shows the how many null values

In [13]:
```
df.isnull()
```

Out[13]:

| | id | gender | age | hypertension | heart_disease | ever_married | work_type | Residence_type | avg_glucose_level | bmi | smoking_status | stroke |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False | False | False | False | True | False | False |
| **2** | False | False | False | False | False | False | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False | False | False | False | False | False | False |
| **4** | False | False | False | False | False | False | False | False | False | False | False | False |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **5105** | False | False | False | False | False | False | False | False | False | True | False | False |
| **5106** | False | False | False | False | False | False | False | False | False | False | False | False |
| **5107** | False | False | False | False | False | False | False | False | False | False | False | False |
| **5108** | False | False | False | False | False | False | False | False | False | False | False | False |
| **5109** | False | False | False | False | False | False | False | False | False | False | False | False |

5110 rows × 12 columns

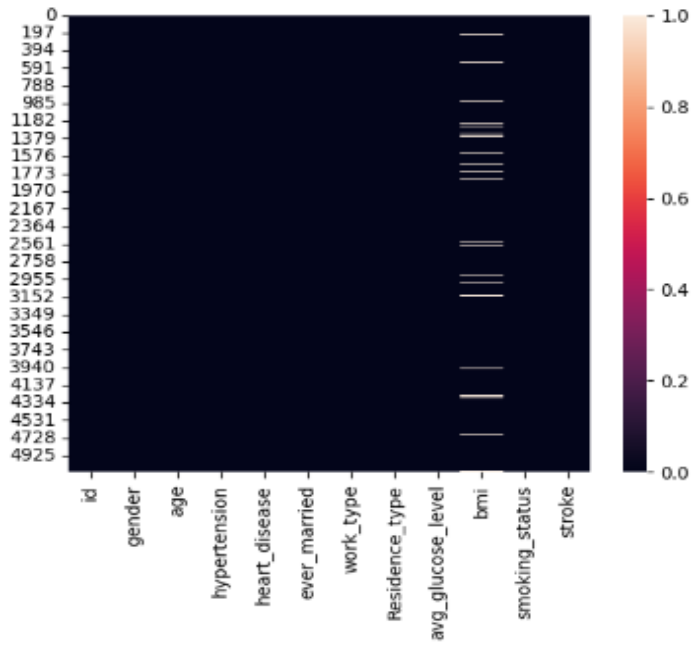
## Is there any Null value present in any Column ? Show with heatmap

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
id                   0
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                201
smoking_status       0
stroke               0
dtype: int64
```

In [16]:
```
sns.heatmap(df.isnull())
```

Out[16]:
```
<Axes: >
```



## Change the Numeric values to categorical

In [17]:
```
#replace 'yes' with 1 and 'no' with 0 in the specified column
df['hypertension'] = df['hypertension'].replace({1: 'yes', 0: 'no'})
```

In [18]:
```
df.dtypes
```

Out[18]:
```
id                     int64
gender                object
age                  float64
hypertension          object
heart_disease          int64
ever_married          object
work_type             object
Residence_type        object
avg_glucose_level    float64
bmi                  float64
smoking_status        object
stroke                 int64
dtype: object
```

In [19]:
```
#replace 'yes' with 1 and 'no' with 0 in the specified column
df['heart_disease'] = df['heart_disease'].replace({1: 'yes', 0: 'no'})
```

In [20]:
```
#replace 'yes' with 1 and 'no' with 0 in the specified column
df['stroke'] = df['stroke'].replace({1: 'yes', 0: 'no'})
```

Bar Graph

In [21]:
```
df.dtypes
```

Out[21]:
```
id                     int64
gender                object
age                  float64
hypertension          object
heart_disease         object
ever_married          object
work_type             object
Residence_type        object
avg_glucose_level    float64
bmi                  float64
smoking_status        object
stroke                object
dtype: object
```

Count of Heart Disease

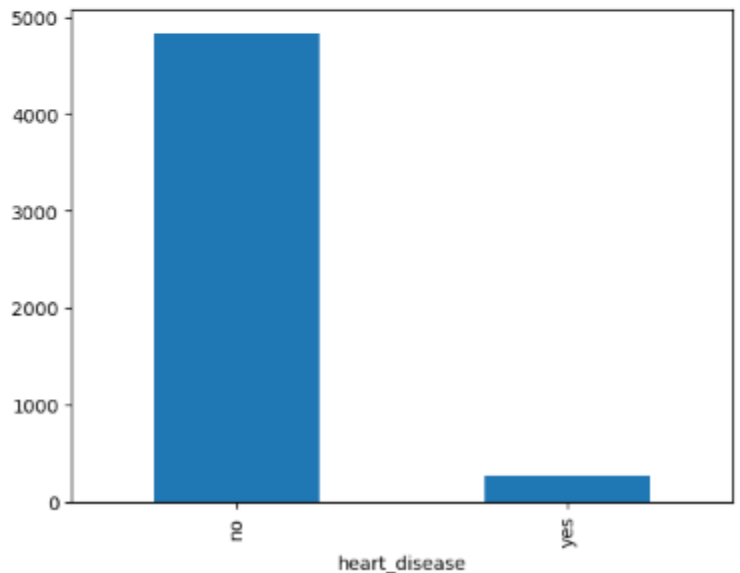
In [23]:
```
ds = df.groupby("heart_disease").heart_disease.count()
```

In [24]:
```
ds
```

Out[24]:
```
heart_disease
no     4834
yes     276
Name: heart_disease, dtype: int64
```

In [25]:
```
ds.plot(kind = 'bar')
```

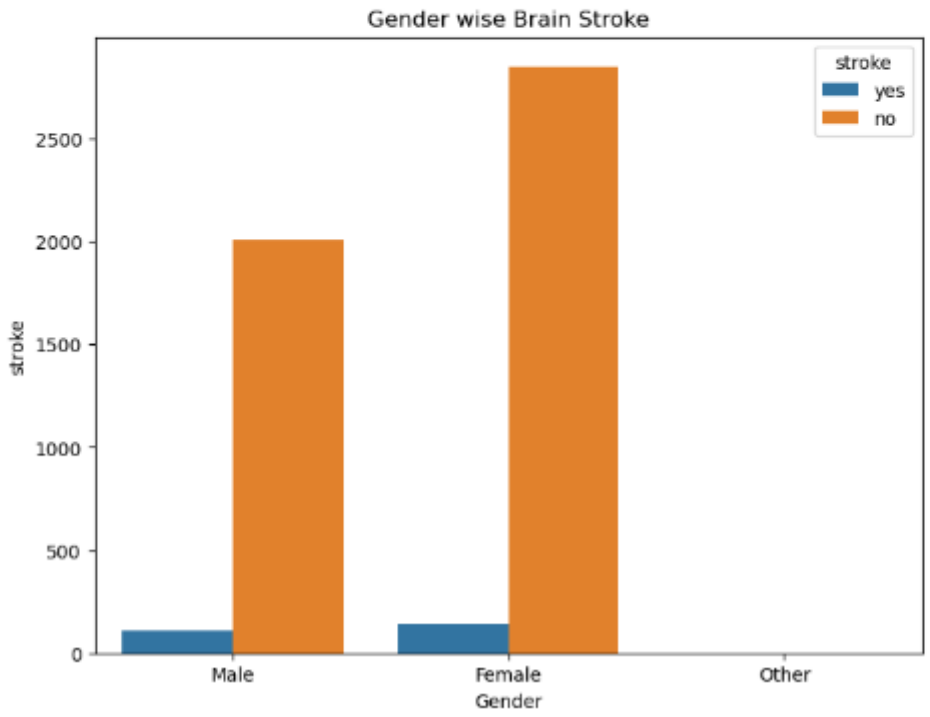
Out[25]:
```
<Axes: xlabel='heart_disease'>
```



In [27]:
```
# Apply conditions and replace values in the column
df['age'] = df['age'].apply(lambda x: 'Younger' if x < 18 else ('Young Adult' if x <= 30 else ('Adult' if 40 <= x <= 50 else 'Se
```

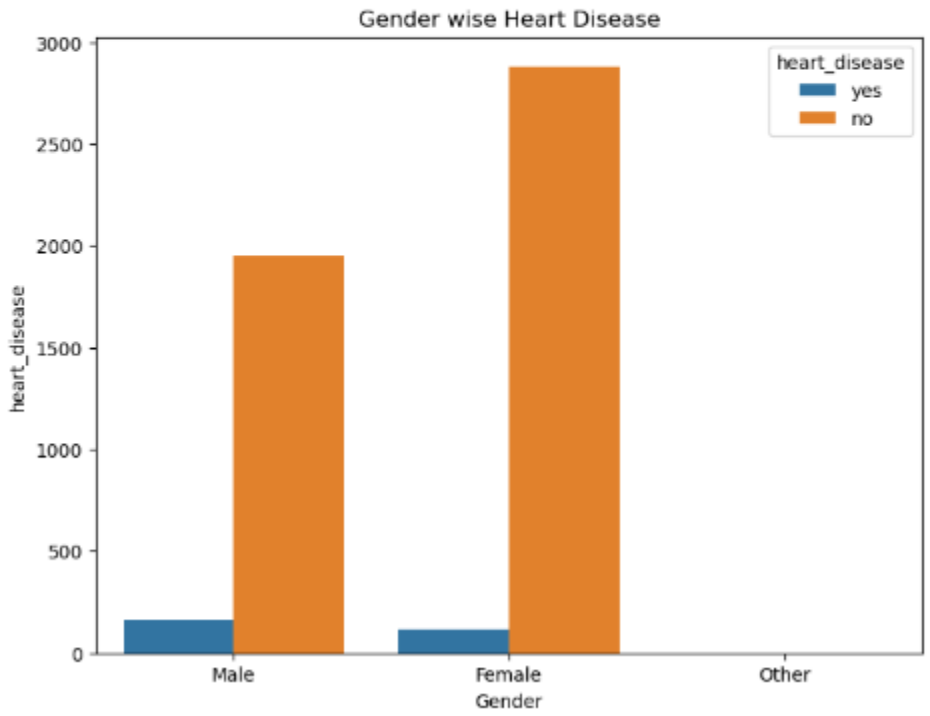
Gender wise Count of Brain Stroke

In [29]:
```
plt.figure(figsize =(8,6))
sns.countplot(data = df, x = 'gender', hue = "stroke")
plt.xlabel('Gender')
plt.ylabel('stroke')
plt.title("Gender wise Brain Stroke")
plt.show()
```



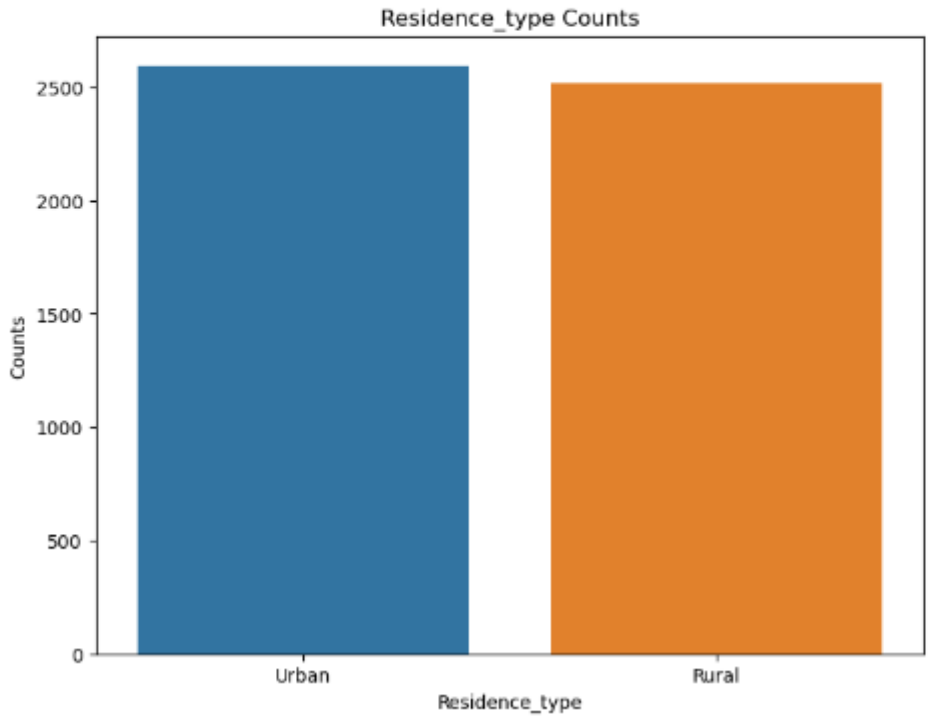
Gender wise Count of Heart Disease

In [30]:
```
plt.figure(figsize = (8,6))
sns.countplot( data = df, x = 'gender', hue = "heart_disease")
plt.xlabel('Gender')
plt.ylabel('heart_disease')
plt.title ('Gender wise Heart Disease')
plt.show()
```



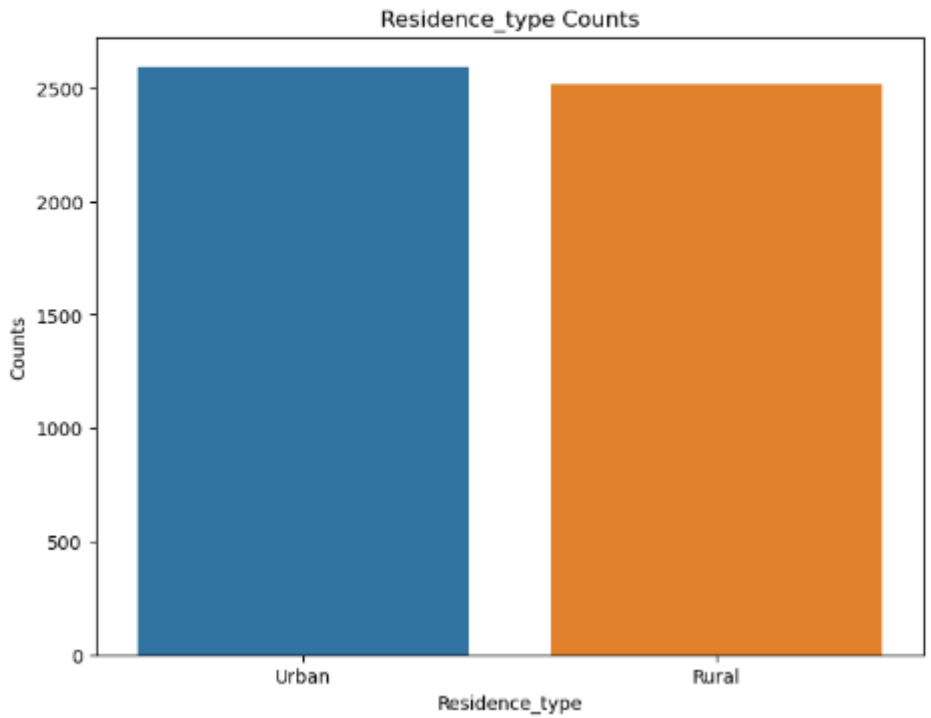
Count of Residency Type

In [31]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Residence_type',)
plt.xlabel('Residence_type')
plt.ylabel('Counts')
plt.title('Residence_type Counts')
plt.show()
```



Count of Residency Type

In [31]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Residence_type',)
plt.xlabel('Residence_type')
plt.ylabel('Counts')
plt.title('Residence_type Counts')
plt.show()
```

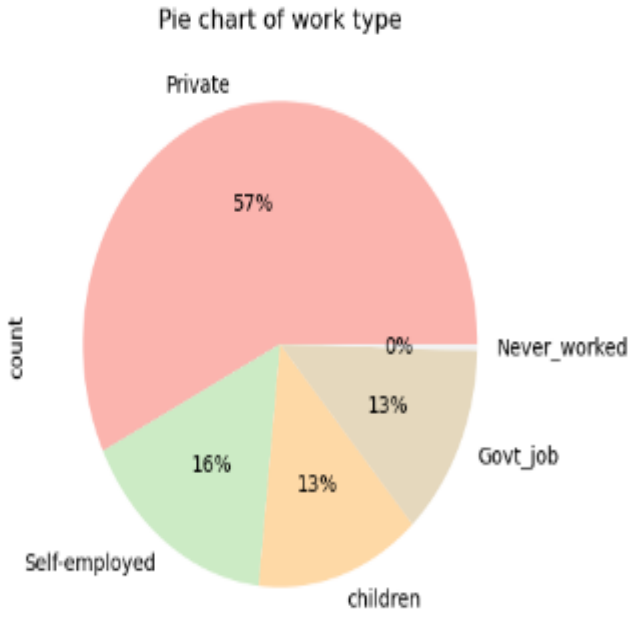


Residence_type

Pie Chart for work Type

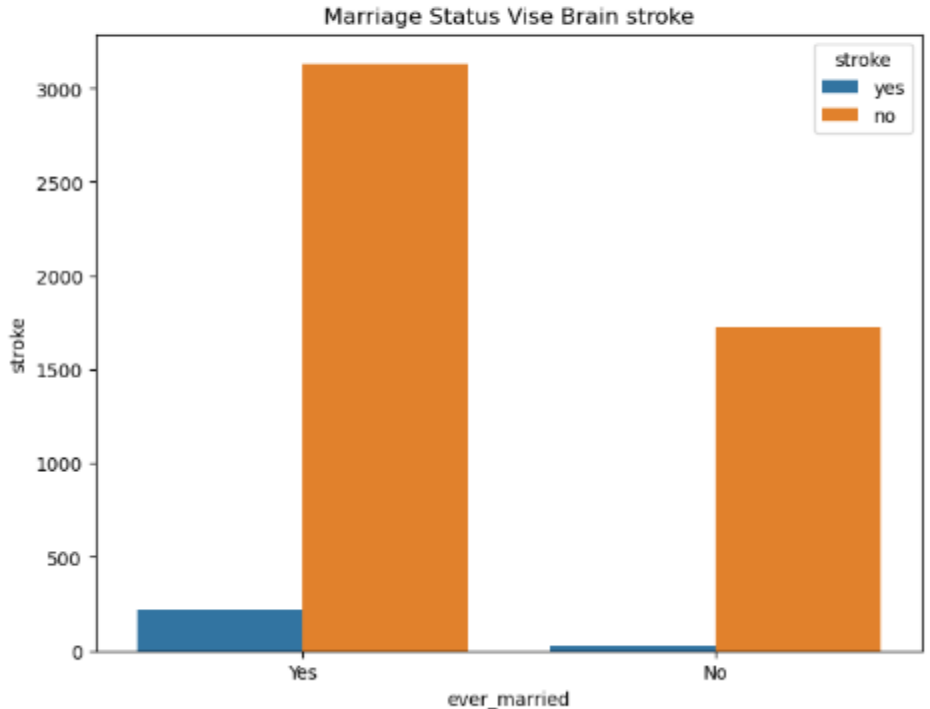
In [32]:
```
df['work_type'].value_counts().plot(kind = 'pie', title = 'Pie chart of work type', autopct="%.0f%%", colormap='Pastel1'),
```

Out[32]:
```
(<Axes: title={'center': 'Pie chart of work type'}, ylabel='count'>,)
```



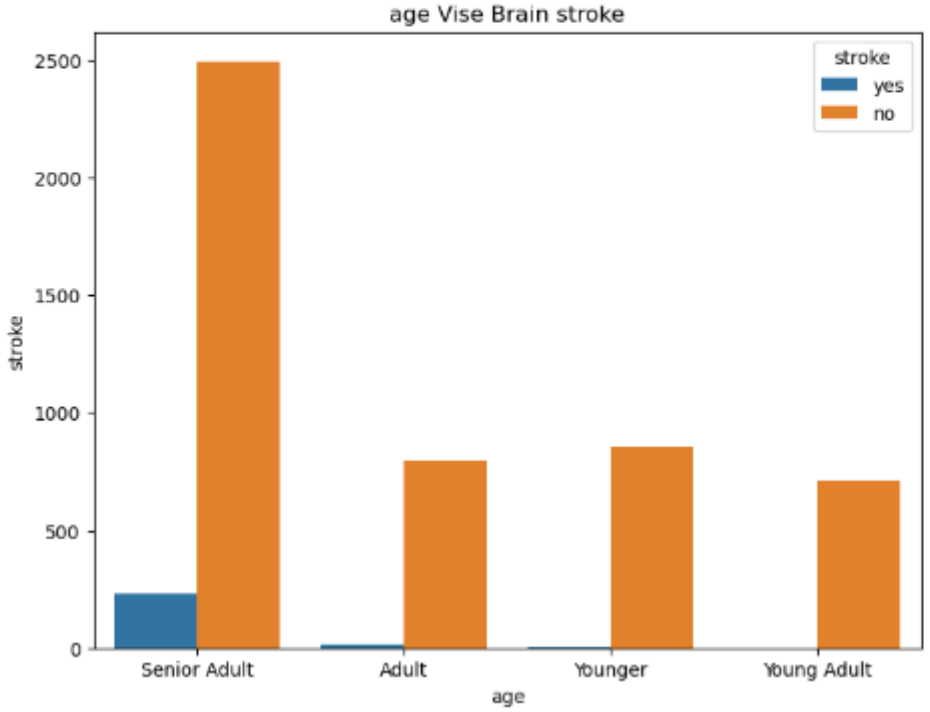
Count Of Marraige wise Brain Stroke

In [33]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='ever_married', hue ="stroke")
plt.xlabel('ever_married')
plt.ylabel('stroke')
plt.title('Marriage Status Vise Brain stroke')
plt.show()
```



Count of Age wise Brain Stroke

In [34]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='age', hue ="stroke")
plt.xlabel('age')
plt.ylabel('stroke')
plt.title('age Vise Brain stroke')
plt.show()
```



Count of Age Wise Smoking

In [35]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='age', hue ="smoking_status")
plt.xlabel('age')
plt.ylabel('smoking_status')
plt.title('Age Vise Smoking Status')
plt.show()
```